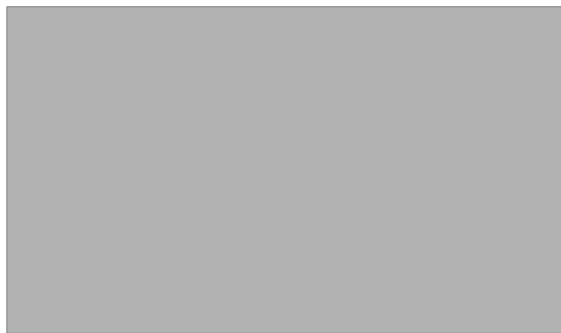# Data Management

Examining the Data

- Look at some random inputs
- Look at the largest and smallest inputs (by file size)
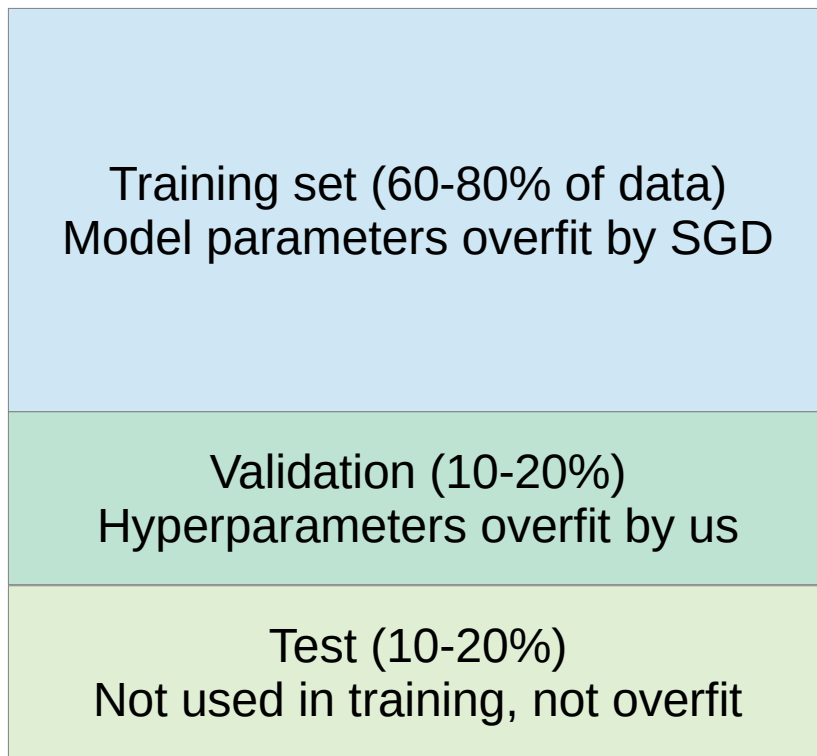 - These are the most likely to be weird or corrupted

Small file

- Manually solve the task

Splitting

- Training set – used by SGD to learn model parameters

- Validation set – used by us to learn hyperparameters

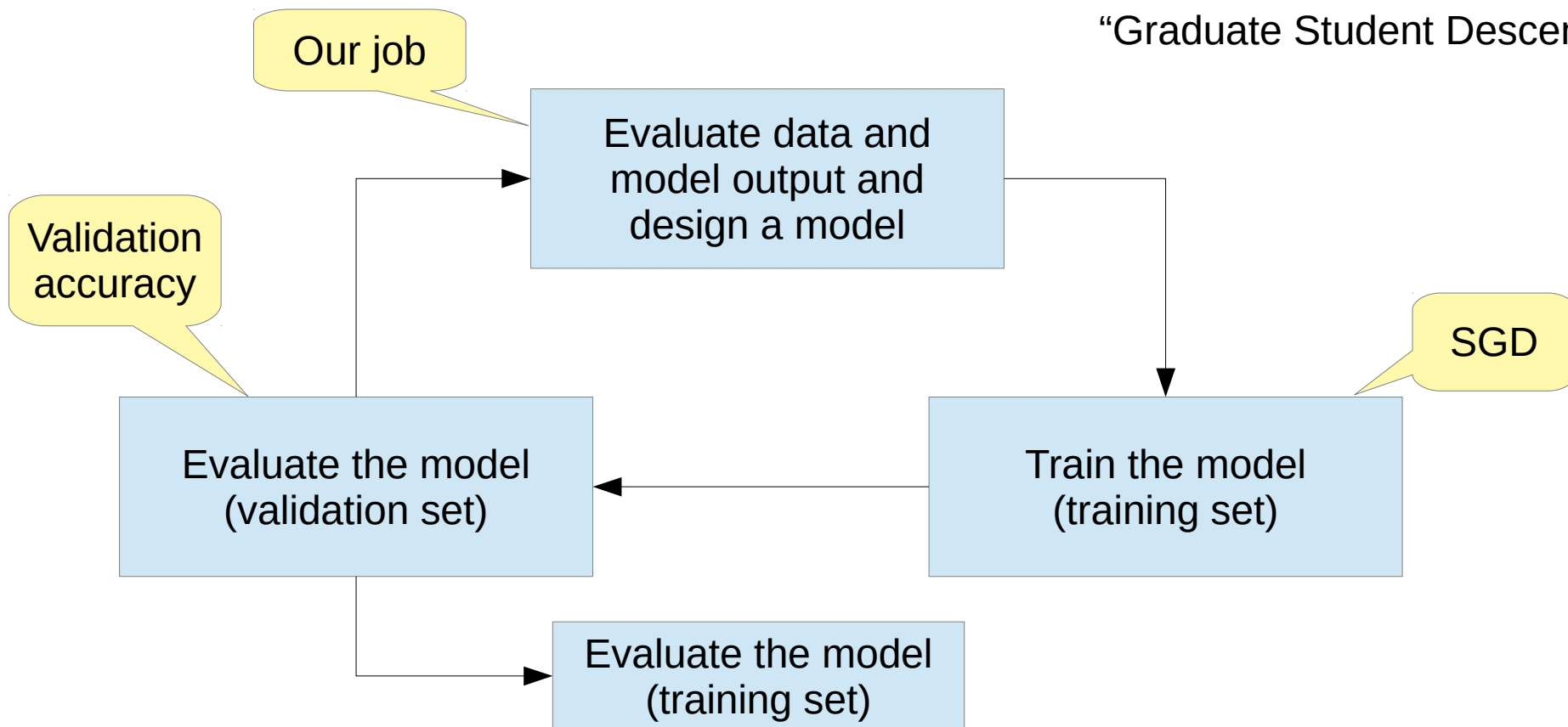- Test set – used once to measure final model performance

Overfitting

- Goal: Learn to classify *unseen* images

- Optimization objective: Learn to classify *training set* images

Training set (60-80% of data)
Model parameters overfit by SGD

Validation (10-20%)
Hyperparameters overfit by us

Test (10-20%)
Not used in training, not overfit

Be sure to split randomly

Tuning

Initialization

Initialization – All Zero

	$x=???$	$g_1=0$	
Linear 1	$v_1=W_1 x$	$g_1=W_1^T g_2$	$\nabla_{W_1} \ell(v_3)=g_2 x^T$
	$v_1=0$	$g_2=0$	0
ReLU	$v_2=\max(v_1, 0)$	$g_2=g_3[v_1>0]$	
	$v_2=0$	$g_3=0$	
Linear 2	$v_3=W_2 v_2$	$g_3=W_2^T g_o$	$\nabla_{W_2} \ell(v_3)=g_o v_2^T$
	$v_3=0$	$g_o=???$	0

$$g_o=\nabla_{v_3} \ell(v_3)$$

Initialization – Random

Normal distribution

$$W_i = N(\mu_i, \sigma_i^2)$$

Means

Standard deviations
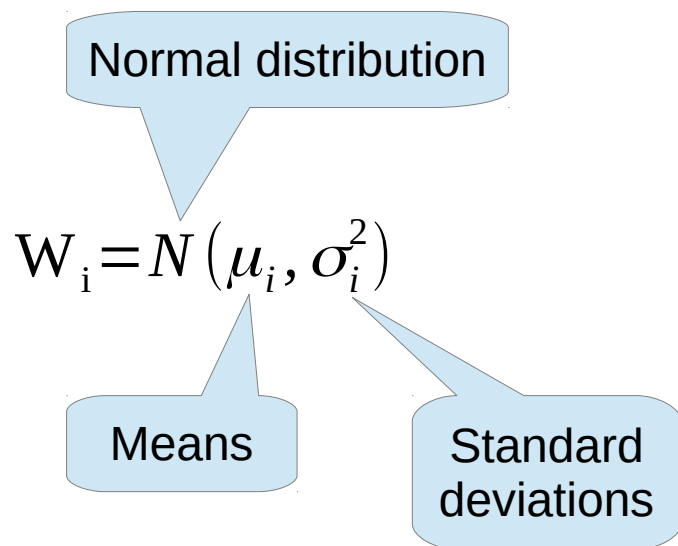
In practice, the mean is zero and we need to choose a standard deviation

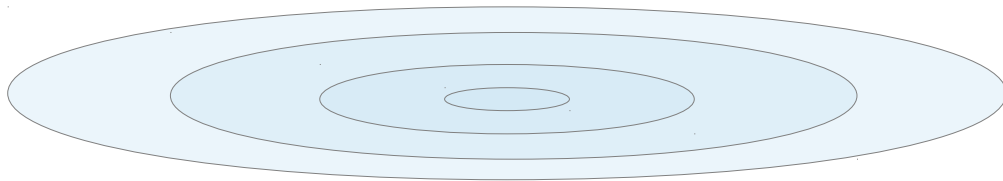# Initialization – Scaling

$$o = W_2 W_1 x$$

$$\nabla_{W_1} \ell(o) = \left(W_2^T \left(\nabla_o \ell(o) \right) \right) x^T$$

$$\nabla_{W_2} \ell(o) = \left(\nabla_o \ell(o) \right) \left(W_1 x \right)^T$$

$$\|W_1\| \approx \|W_2\|$$
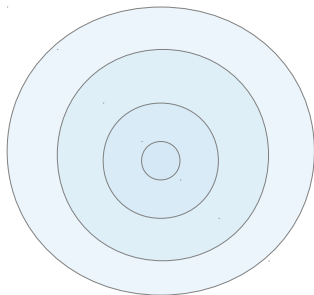
$$\|W_1\| \gg \|W_2\|$$

Choosing a Standard Deviation

Math

(See the supplementary material for details)

Kaiming Initialization

- Choose either activations or gradients and keep the magnitude roughly constant:

  - Activations: $W \sim N\left(0, \dfrac{2}{n_{\text{in}}}\right)$

  - Gradients: $W \sim N\left(0, \dfrac{2}{n_{\text{out}}}\right)$

Xavier Initialization

- Keep both activation and gradient magnitudes roughly constant throughout the network

$$W \sim N\left(0, \frac{4}{n_{\text{in}} + n_{\text{out}}}\right)$$

In Practice

- Xavier initialization (the PyTorch default) is usually good enough.

- The last layer can be initialized to zero.
  - If the previous layers are not zero, the last activation is not zero, so we still get gradients.